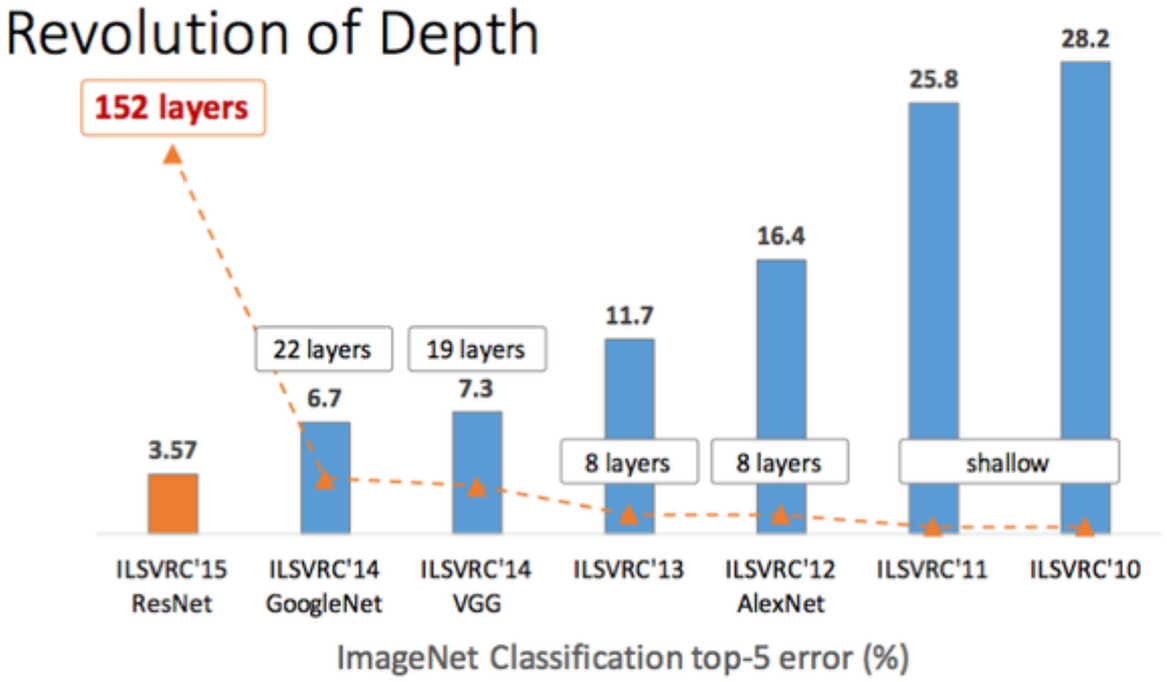
ResNet

Deep Residual Learning for Image Recognition

2022.07.13 한나연

VGGNet (2014)



- Convolution layer를 깊이 쌓을수록 성능이 증가
- 그럼 더 깊게 쌓아보자!

VGGNet Architecture

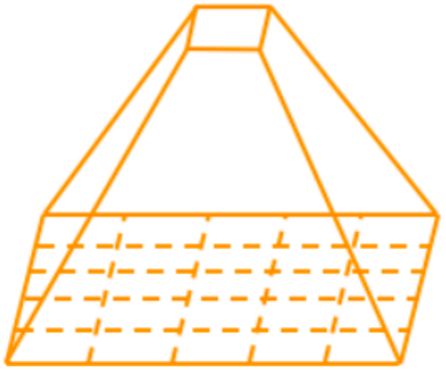
Table 3: **ConvNet performance at a single test scale.**

ConvNet config. (Table 1)	smallest image side		top-1 val. error (%)	top-5 val. error (%)
	train (S)	test (Q)		
A	256	256	29.6	10.4
A-LRN	256	256	29.7	10.5
B	256	256	28.7	9.9
C	256	256	28.1	9.4
	384	384	28.1	9.3
	[256;512]	384	27.3	8.8
D	256	256	27.0	8.8
	384	384	26.8	8.7
	[256;512]	384	25.6	8.1
E	256	256	27.3	9.0
	384	384	26.9	8.7
	[256;512]	384	**25.5**	**8.0**

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

- 작은 크기의 3x3 필터 사용
- Data augmentation: random crop, 다양한 크기의 입력 이미지(256~512)
- convolution layer를 연속적으로 11-19개 쌓음
  - layer가 깊어질수록 error 감소
  - 다양한 scale로 resize한 이미지를 사용했을 때 더 높은 성능을 보임

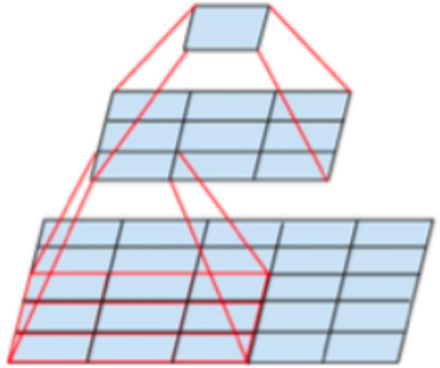
VGGNet: 작은 필터를 사용하자



5x5 convolution

5*5 필터 1번 사용

파라미터 수 = 5*5 = 25



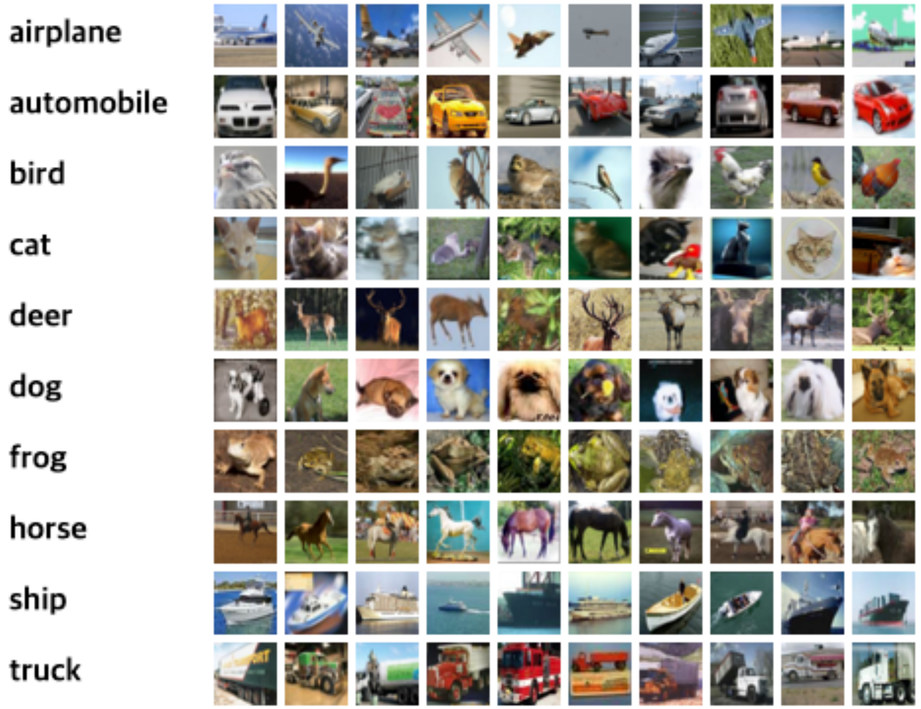
two successive
3x3 convolutions

3*3 필터 2번 사용

파라미터 수 =(3*3)*2 = 18

- 적은 파라미터로 동일한 receptive field의 특징 추출 가능
- non-linear activation function을 더 사용해 더 다양한 특징을 추출할 수 있음(비선형성 부여)
- 비선형성(non-linearity)이 왜 필요한가?
  - 데이터가 복잡해지고, 특징의 차원이 증가하면서 데이터의 분포가 선형적이지 않고 비선형적으로 나타남
  - 이러한 데이터에서는 비선형 boundary로 표현이 가능하기에, 비선형성이 필요

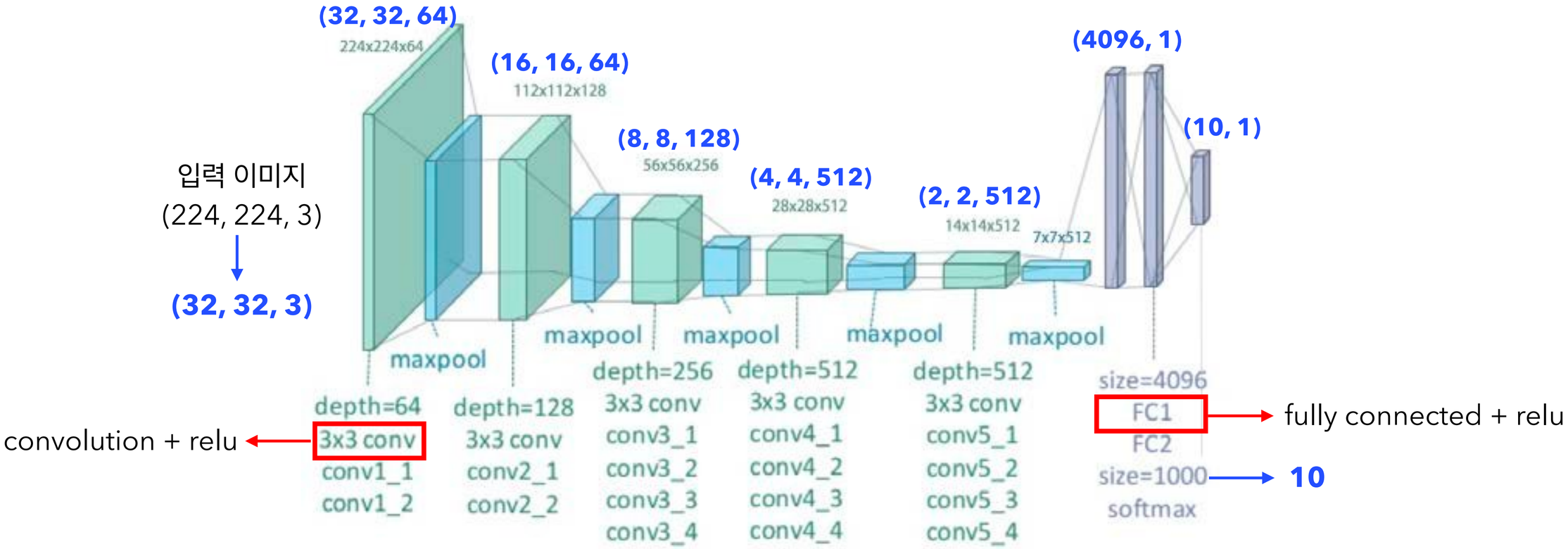
CIFAR-10 데이터셋



- 32x32 크기의 컬러 이미지 6만장 (train 5만장, test 1만장)
- 클래스 10개
- 이미지 분류 데이터셋
  - MNIST: 손글씨 데이터, 클래스 10개
  - ImageNet: 클래스 1000개
  - CIFAR-10: torchvision.datasets에서 쉽게 다운로드 가능

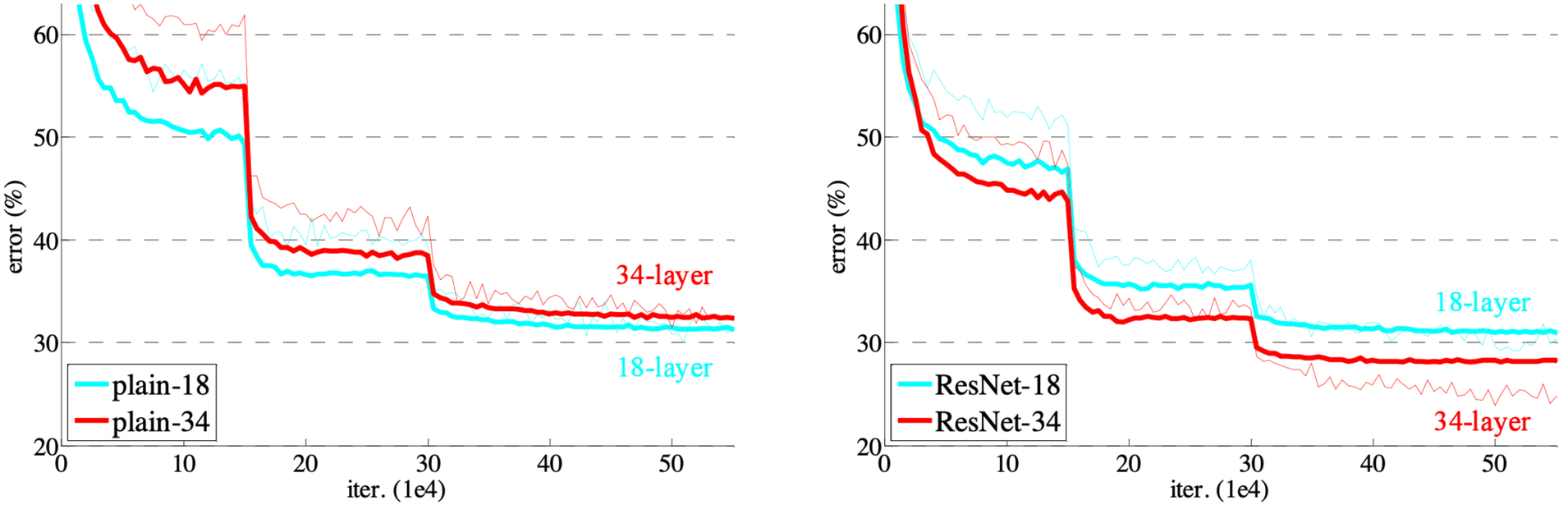
- CIFAR-10: https://www.cs.toronto.edu/~kriz/cifar.html
- torchvision.datasets.CIFAR10: https://pytorch.org/vision/stable/generated/torchvision.datasets.CIFAR10.html#torchvision.datasets.CIFAR10

VGG-19 구현



- Max pooling시 이미지 크기가 절반으로 줄어듦
- 파란색 글씨는 CIFAR=10 데이터셋을 사용해 달라진 부분

ResNet (2015)

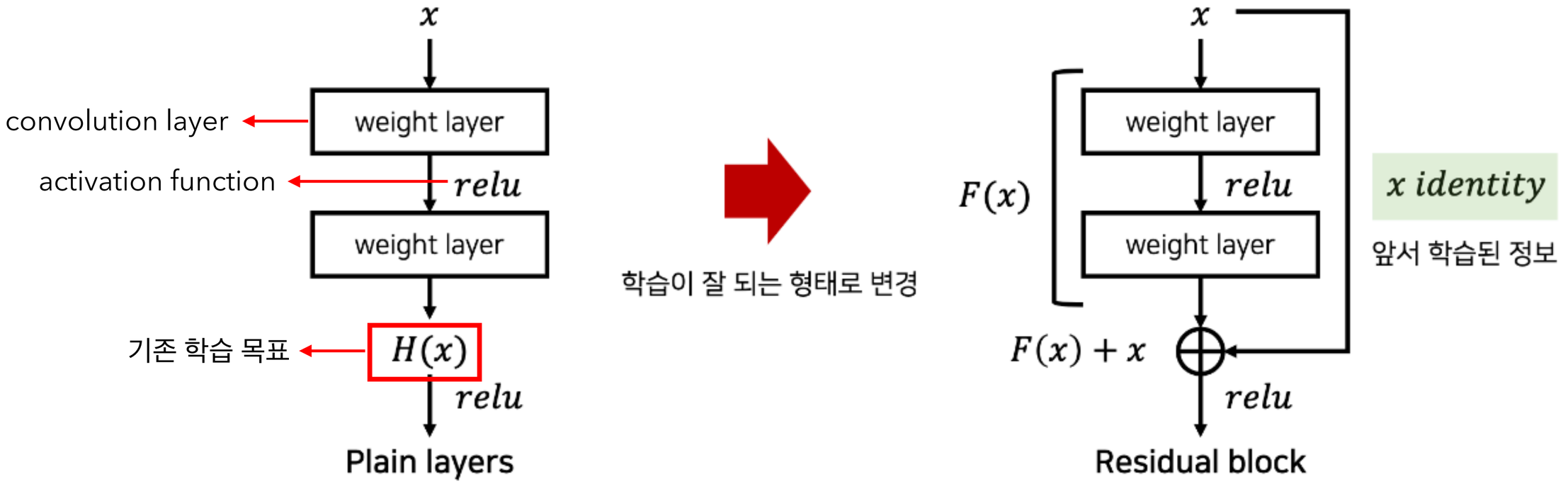


- 단순히 convolution layer를 깊게 쌓는 방법은 일정 layer보다 깊어지면 성능을 떨어트림
- Residual block을 사용해 학습 난이도를 낮추자!

ResNet

$$H(x) = F(x) + x$$

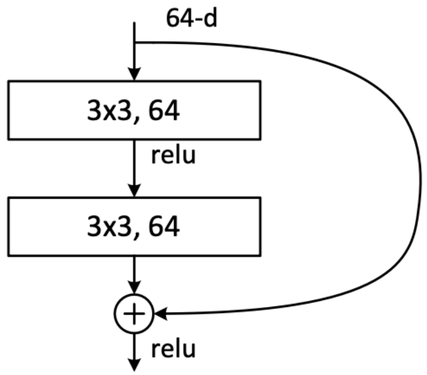
$$F(x) = H(x) - x$$



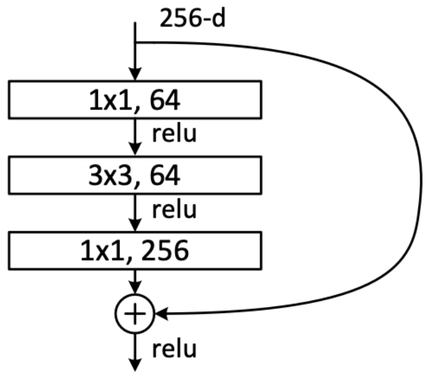
- 입력(x)과 컨볼루션 연산 결과의 차이점만 학습하자
- 수식적으로는 동일하지만, 모델 학습 관점에서는 $H(x)$보다 $H(x) - x$가 더 학습하기 수월함

ResNet Architecture

layer name	output size	18-layer	34-layer	50-layer
conv1	112×112			7×7, 64, stride 2
				3×3 max pool, stride 2
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softr		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9

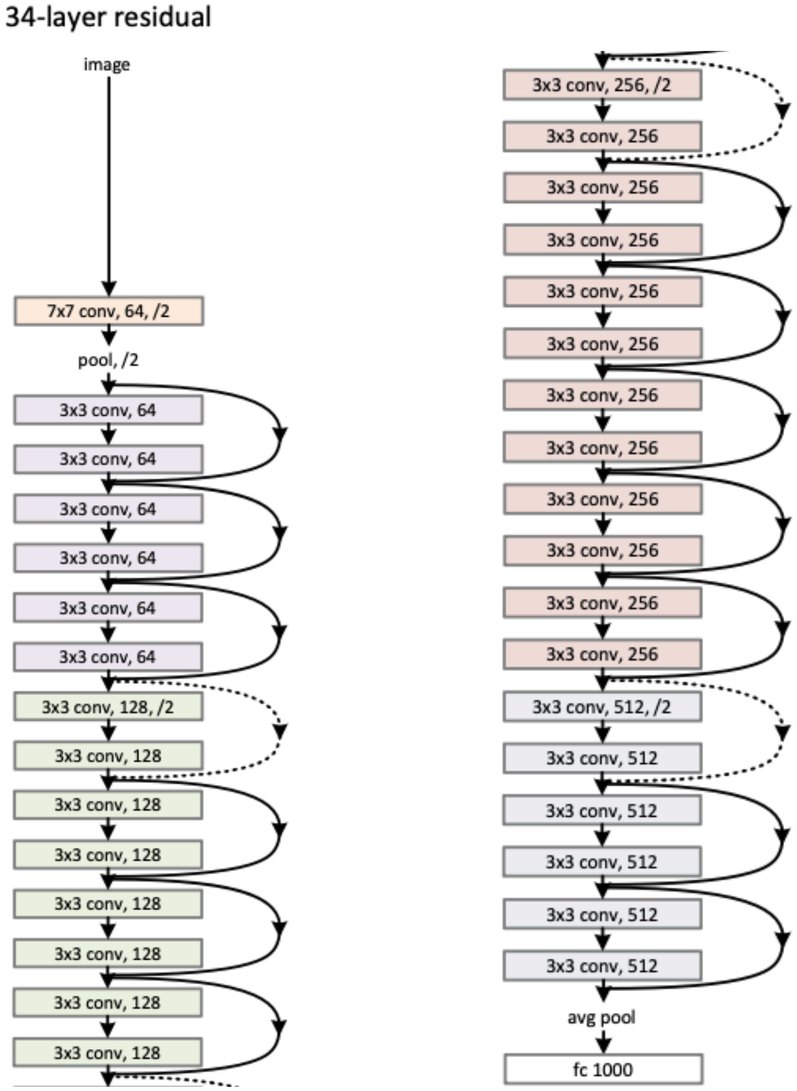


ResNet-18, 34



ResNet-50 이상

ResNet


34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

ResNet

3.3. Network Architectures

We have tested various plain/residual nets, and have observed consistent phenomena. To provide instances for discussion, we describe two models for ImageNet as follows.

Plain Network. Our plain baselines (Fig. 3, middle) are mainly inspired by the philosophy of VGG nets [41] (Fig. 3, left). The convolutional layers mostly have 3×3 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax. The total number of weighted layers is 34 in Fig. 3 (middle).

It is worth noticing that our model has *fewer* filters and *lower* complexity than VGG nets [41] (Fig. 3, left). Our 34-layer baseline has 3.6 billion FLOPs (multiply-adds), which is only 18% of VGG-19 (19.6 billion FLOPs).

- 3x3 filter 사용
- Convolution layer 직후에 stride=2로 downsampling
- 마지막에 average pooling layer -> (num_classes)차원의 fully connected layer -> softmax